strlcpy() and strlcat()

Daniel Plakosh, Software Engineering Institute [vita[1]]

2006-01-30

The standard C library includes functions that are designed to prevent buffer overflows, particularly `strncpy()` and `strncat()`. These universally available functions discard data larger than the specified length, regardless of whether it fits into the buffer. These functions are deprecated for new Windows code because they are frequently used incorrectly.

Development Context

Copying and concatenating character strings

Technology Context

C, UNIX, FreeBSD, OpenBSD, NetBSD, MacOS X, Solaris

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Improper use of the `strncpy()` and `strncat()` functions can result in buffer overflow vulnerabilities.

Description

Many UNIX variants provide the `strlcpy()` and `strlcat()` functions to copy and concatenate strings in a less error-prone manner. These functions' prototypes are as follows:

```
size_t strlcpy(char *dst, const char *src, size_t size);
```

```
size_t strlcat(char *dst, const char *src, size_t size);
```

The `strlcat()` function copies the null-terminated string from src to dst (up to `size` characters). The `strlcat()` function appends the null-terminated string src to the end of dst (but no more than `size` characters will be in the destination).

1. daisy:268 (Plakosh, Daniel)

To help prevent writing outside the bounds of the array, the `strlcpy()` and `strlcat()` functions accept the full size of the destination string as a size parameter. For statically allocated buffers, this value is easily computed at compile time using the `sizeof()` operator.

Both functions guarantee that the destination string is null terminated for all non-zero-length buffers to prevent null-termination errors.

The `strlcpy()` and `strlcat()` functions return the total length of the string created. For `strlcpy()` that is simply the length of the source; for `strlcat()` it is the length of the destination (before concatenation) plus the length of the source. To check for truncation, the programmer need only verify that the return value is less than the size parameter. If the resulting string is truncated, the programmer now knows the number of bytes needed to store the entire string and may reallocate and recopy. This helps prevent errors resulting from an unintentional loss of data.

Neither `strlcpy()` nor `strlcat()` zero-fill their destination strings (other than the compulsory null byte to terminate the string). This results in performance close to that of `strcpy()` and much better than `strncpy()` [ISO/IEC 99]. Table 1 shows the elapsed time required to copy the string “this is just a test” 1000 times into a 1024 byte buffer [Miller 99].

Table 1. Performance in seconds

CPU	Function	Time (sec.)
mk68	strcpy()	0.137
mk68	strncpy()	0.464
mk68	strlcpy()	0.140
alpha	strcpy()	0.018
alpha	strncpy()	0.100
alpha	strlcpy()	0.020

Unfortunately, `strlcpy()` and `strlcat()` are not universally available in the standard libraries of UNIX systems. Both functions are defined in `string.h` for many UNIX variants, including OpenBSD and Solaris, but not for GNU/Linux. Because these are relatively small functions, however, you can easily include them in your own program's source whenever the underlying system doesn't provide them.

References

[ISO/IEC 99] ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming languages — C*. International Organization for Standardization, 1999.

[Miller 99] Miller, T. C. & de Raadt, T. "strlcpy and strlcat—Consistent, Safe String Copy and Concatenation," 175-178. *Proceedings of the FREENIX Track, 1999 USENIX Annual Technical Conference*. Monterey, CA, June 6-11, 1999. Berkeley, CA: USENIX Association, 1999. http://www.usenix.org/publications/library/proceedings/usenix9 full_papers/millert/millert.pdf.

Pearson Education, Inc. Copyright

Velden

Naam	Waarde
Copyright Holder	Pearson Education

Velden

Naam	Waarde
is-content-area-overview	false
Content Areas	Knowledge/Coding Practices
SDLC Relevance	Implementation
Workflow State	Publishable